Final project: Report

Lucas Nesi

1 Report

The QR factorization of sparse matrices using the software package `QR_mumps` may present different behaviors regarding the selection of execution parameters for the StarPU runtime and the own `QR_mumps`. An erroneous selection of parameters values can negatively impact in the application performance, while a correct configuration could provide positive impacts using the same workload and hardware. The problem of parameter tuning is present in many applications usually because they represent options in heuristic parts of the application, like the StarPU scheduler, or in arbitrary constants, like the block size used for the matrix decomposition. The objectives of this report are the following. (i) For a set of selected parameters, indicate which ones had the most impact on the performance. (ii) Considering the selected options for each parameter, find the best and worst combination.

The chosen parameters for this study as factors are the following.

- (i) StarPUs Scheduling, with levels `LWS` and `DMDA`, where the `LWS` is a simpler scheduler while the `DMDA` is a more sophisticated one; presenting very different options to evaluate this factor.
- (ii) StarPUs number of workers per GPU, from 0 to 4, the possible values for this parameter.
- (iii) `QR_mumps` block size, with levels 320, 640 and 960, common values used in many algebraic kernels.
- (iv) `QR_mumps` inner block size, with levels 32 and 64, this parameter requires to be a divider of the block size.

The input matrices used in this study are present in Table 1.

Table 1: Selected matrices for workload

Matrix	Rows	Columns	Non-Zeros
TF16	15437	19321	216173
TF17	38132	48630	586218
CH7-7-b5	35280	52920	211680
MK12-b3	51975	13860	207900

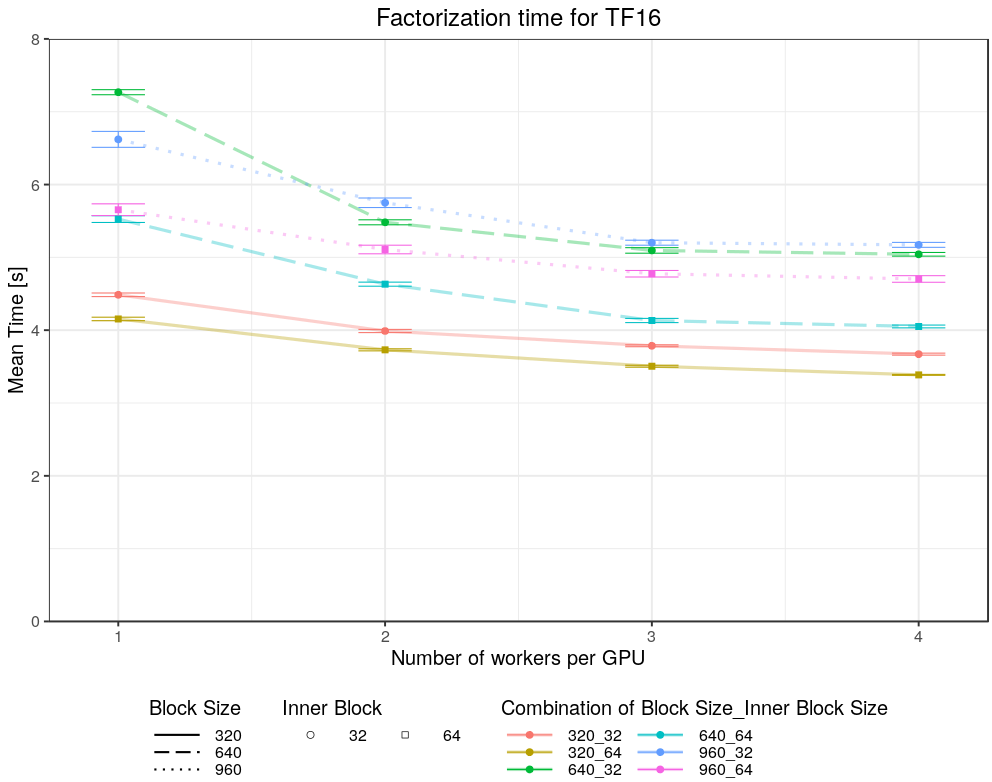
First we conducted a $2^k n$ experiment setup to find the most impacting parameters. We construct a allocation of variation table following the guidelines of Raj Jain, in his book The art of computer systems performance analysis, chapter 18. The resultant table is

The data indicates that the factor A (scheduling) had almost none impact on all conditions, including all combinations with it. The factor B (use of GPU) had the most performance influence on all the circumstances, reaching 95.26% using the TF17 on the `TUPI` case. The C factor (block size) presented a small impact on TF17 `Draco` case and almost none on the other two, yet its combination with B, the BC case, presented small variations on all situations. The last isolated factor, D, the inner block size, exhibited medium allocation

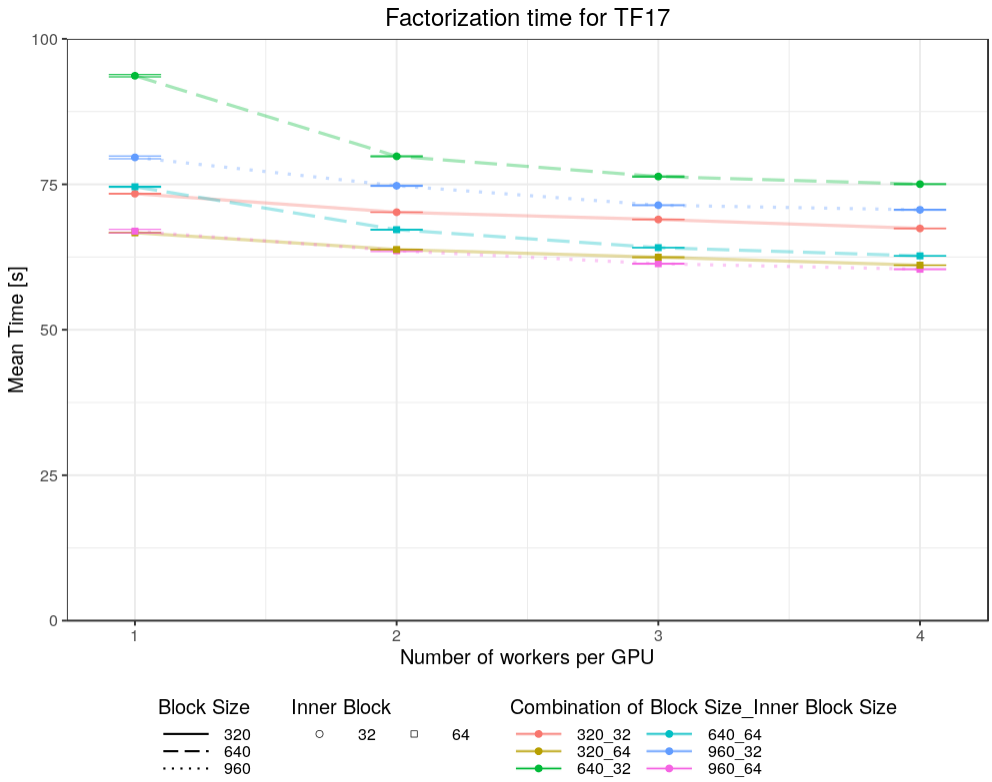
Factor	A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	ABCD
TF17 draco	0.02%	83.59%	3.62%	9.36%	0.0%	0.0%	0.0%	3.13%	0.15%	0.07%	0.00%	0.00%	0.00%	0.05%	0.00%
TF17 tupi	0.00%	95.26%	0.13%	2.93%	0.0%	0.0%	0.0%	1.34%	0.10%	0.14%	0.00%	0.00%	0.00%	0.10%	0.00%
TF16 draco	0.00%	92.88%	0.27%	2.95%	0.0%	0.0%	0.0%	3.52%	0.23%	0.03%	0.00%	0.00%	0.00%	0.10%	0.01%

of variation on the first case and a modest on the other two. All the additional combinations presented a minimal allocation of variance.

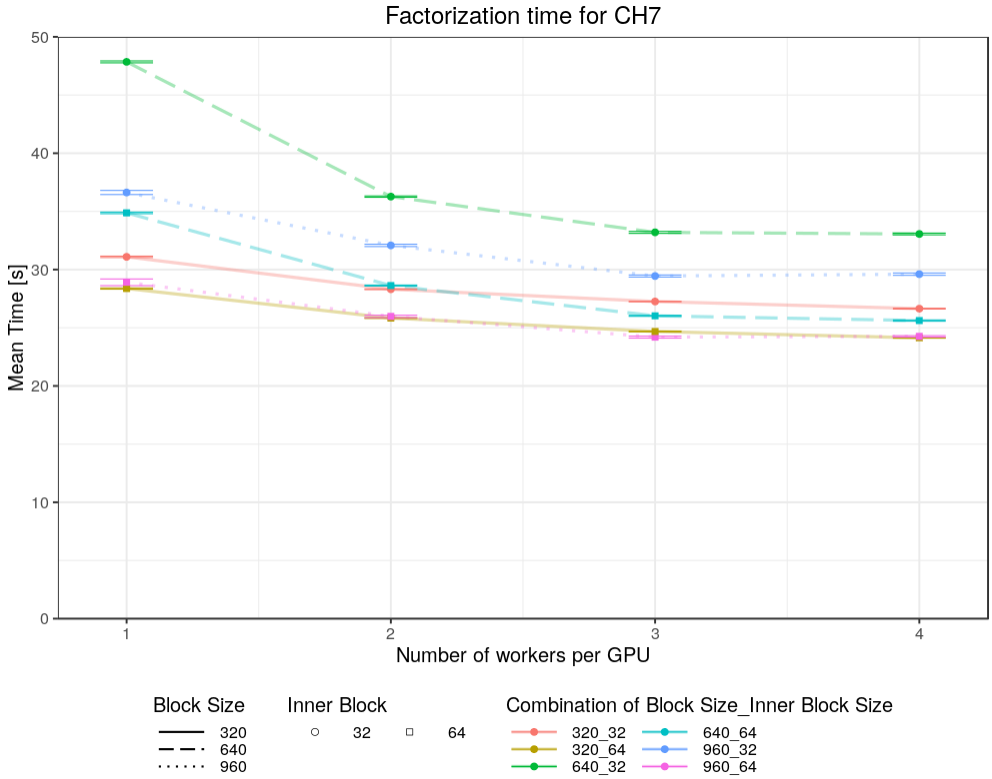
After the first step, we conclude that the scheduler parameter had almost no impact, so we decided to concentrate the next experiment on the other factors. Also, after we checked that the use of GPU was the most impacting factor, we executed all the subsequent tests with at least one worker per GPU, removing the level of zero workers (not using GPUs). The parameters for the next batch of experiments are the following. (i) Workers per GPU, from one to four; (ii) Block size, with levels 360, 640 and 960; (iii) Inner block size, with values 32 and 64. Moreover, to reduce the number of experiments, and because we didnt notice any drastic difference between machines, we executed all the next experiments on the `TUPI` machine. Each combination is performed 30 times with the sequence between configurations randomized. We generate Figures to report the times for each configuration. In the X-axis there is the number of workers per GPU, and in the Y-axis the factorization time means, with a confidence of 99%. Also, the circle points represent the use of 32 as the inner block size and the squares the value 64. The line shape indicates the value for the block size; where the continuous line is 320, the dashed is 640, and the dotted is 960. Also, the colors reinforce the combination of block size and the inner block size (colors makes easier to check the block size and the inner block size combination). The Figure for the TF16:



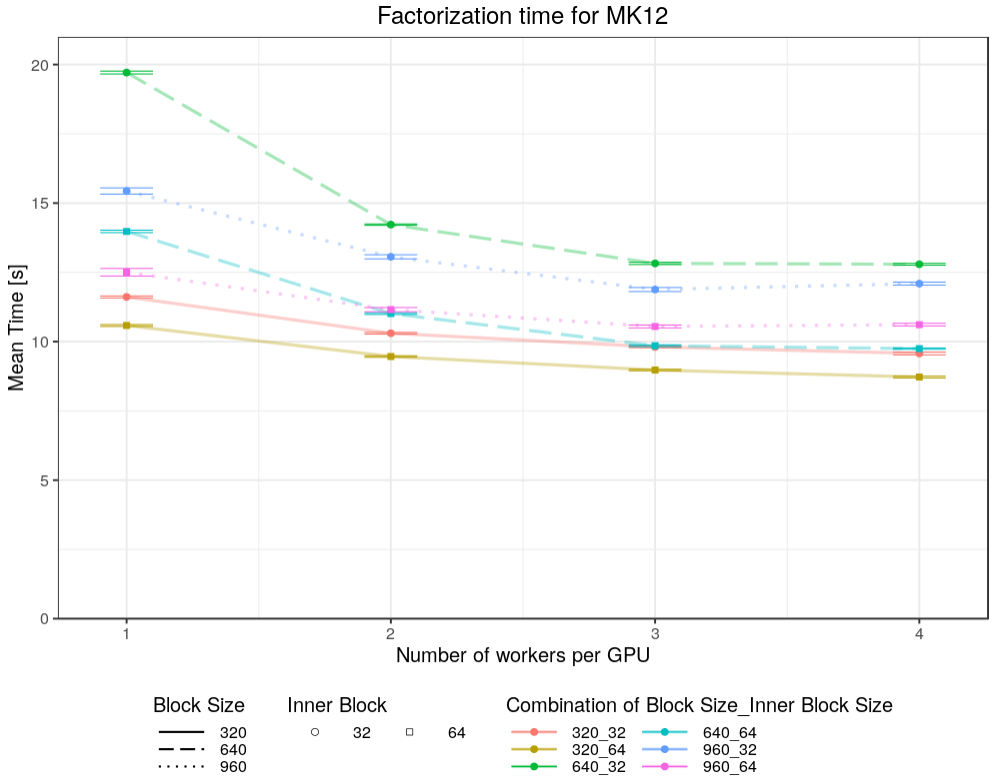
We can check the mean factorization time results for the execution using the TF16 matrix. The combination 320 64 is the best one independent (with statistic significance) of the number of GPUs. Also, there is some performance increase from one worker per GPU to two workers, and then three; however, when increasing to four workers the minimal difference is inside the confidence interval. The Figure for the TF17:



Similar to the TF16 case, the combination 320 64 was the best one independent of the number of GPU workers. However, in this case, it shared the position with the combination 960 64 that had a statistically tied using the confidence interval. Moreover, the worst combinations, in this case, is using the inner block size of 32. The performance difference when increasing the number of workers per GPU is more apparent from one to two workers, while from 3 to 4 none significant statistical difference in performance is noted. The Figure for the CH7:



This case has similar results to the TF17 one. It worth to mention that these matrices were the larges ones and also had the biggest execution time among the tested ones. The combinations 320 64 and 960 64 were the best ones, in a statistical tie, independent of the number of workers per GPU. The worst cases were the three combinations using the inner block size of 32, and there is a significant performance increase from one to two workers per GPU. The Figure for the MK12:



It had similar results with the TF16 matrix, both ones were the smallest matrices and had the lower execution times from the tested ones. The case 320 64 was the best one alone while the 640 32 was the worst one in all cases independent of the number of workers per GPU.

2 Conclusion and Next Steps

In this investigation, we studied a set of execution parameters of the software package `QR_Mumps` that run over the StarPU runtime. We selected four parameters, two from each software to understand how much they change the final application performance. In our experiments, we verified that some of the parameters had much more impact than others, and report the best values for each one.

From the four select parameters, StarPU scheduler, StarPU number of workers per GPU, `QR_Mumps` block size, and `QR_Mumps` inner block size, we verified the individual impact using the allocation of variation method. We concluded that the StarPU scheduler did not influence the performance, while the StarPU number of workers per GPU had the most impact. The block size and the inner block size had some effect, where different select values presented statistical difference. For this reason, we not considered the StarPU scheduler in the next experimental step.

After the first step of experiments, we evaluate all the options for the selected three factors; Number of Workers per GPU, Block Size, and inner block size. The behavior of the parameter number of workers per GPU indicates that the GPUs got saturated after two workers, however, adding more workers didnt impact negatively in the performance, any value greater than two is sufficient here. Considering the block size, the only option that was the best regarding any matrix was 320. In all cases, the inner block size of 64 was better than the equivalent 32 variant. In conclusion, the best combination of the proposed parameters is any scheduler, four workers per GPU, block size of 320, and inner block size of 64.

In future work, we intend to investigate why these behaviors occur. This goal could be achieved with the use of executions traces and performance analysis toolkits like StarVZ. Also the investigation of other parameters of both StarPU and `QR_Mumps` is also possible to continue to improve the performance of the application without changing the hardware or the source code.